SOFTWARE ENGINEERING

SUBMITTED TO: LEC. SAIMA YASMEEN

SUBMITTED BY: ISRA SAAD

SAMIA REHMAN

M. ABUBAKAR

EESA KHURRAM

PBL

NATIONAL UNIVERSITY OF TECHNOLOGY, ISLAMABAD.

Software Requirements Specification (SRS)

Project Title

Gesture-Based Mouse and Volume Control System using Computer Vision

---

1. Introduction

1.1 Problem Identification

In modern computing environments, interaction with computers is still largely dependent on physical input devices such as a mouse, keyboard, or dedicated hardware buttons for volume control. These traditional devices can be inconvenient or inaccessible in scenarios such as presentations, multimedia control, hygiene-sensitive environments, or for users with physical disabilities.

The problem addressed by this project is the lack of a natural, touch-free, and intuitive way to control basic computer operations like mouse movement, clicking, and audio volume. This issue affects general users, presenters, content creators, and individuals with limited motor abilities. A software-based, vision-driven solution is required to reduce dependency on physical hardware and enable human–computer interaction through hand gestures.

1.2 Purpose of Selecting This Project

This project was selected due to its strong relevance to real-world human–computer interaction problems and its practical application of Software Engineering principles. It integrates concepts of computer vision, artificial intelligence, and system design while solving a meaningful accessibility and usability problem. The project also provides hands-on experience with requirements analysis, design modeling, implementation, and testing.

1.3 Project Description

The Gesture-Based Mouse and Volume Control System is a desktop application that uses a webcam and computer vision techniques to detect hand gestures and translate them into system actions. The left hand controls mouse movement and clicks, while the right hand controls system volume and mute/unmute functionality. The system uses MediaPipe for hand tracking, OpenCV for image processing, PyAutoGUI for mouse control, and PyCaw for audio control.

1.4 Project Scope

In Scope:

- Real-time hand detection using a webcam
- Mouse cursor movement using hand gestures
- Left mouse click using finger bending gesture
- System volume control using pinch gesture
- Mute and unmute functionality using palm and fist gestures
- Visual feedback on screen

Out of Scope:

- Gesture-based keyboard input
- Multi-monitor support
- Custom gesture training
- Mobile platform support

Limitations:

- Requires a functional webcam
- Performance depends on lighting conditions
- Designed primarily for Windows OS

1.5 Target Users / Stakeholders

- General computer users
- Students and presenters
- Content creators
- Users with physical disabilities

- Instructor and evaluators

2. Requirements Specification

2.1 Functional Requirements (FRs)

- **FR-01:** The system shall detect hands in real time using a webcam.
- **FR-02:** The system shall track left and right hands separately.
- **FR-03:** The system shall move the mouse cursor based on left-hand movement.
- **FR-04:** The system shall perform a left mouse click when the index finger bends.
- **FR-05:** The system shall detect a right-hand pinch gesture.
- **FR-06:** The system shall adjust system volume based on pinch distance.
- **FR-07:** The system shall mute audio when an open palm gesture is detected.
- **FR-08:** The system shall unmute audio when a fist gesture is detected.
- **FR-09:** The system shall display visual feedback for gestures.
- **FR-10:** The system shall allow the user to exit the application using a key press.

2.2 Non-Functional Requirements (NFRs)

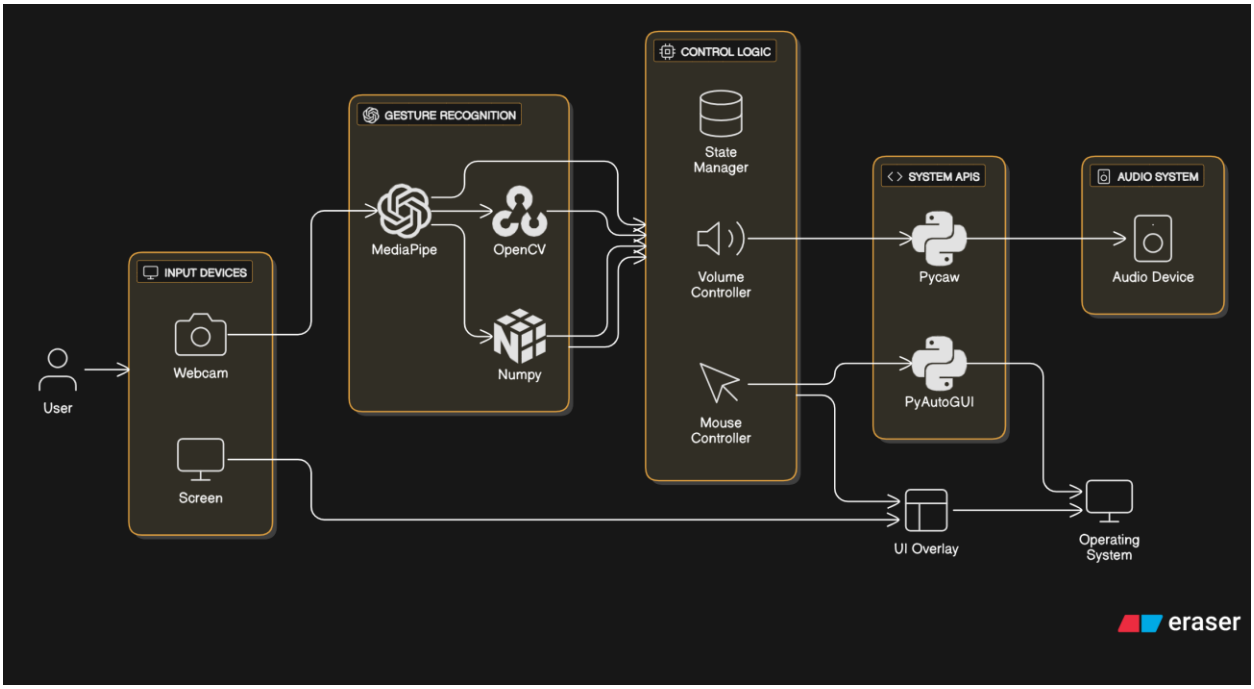
- **NFR-01 (Performance):** The system shall respond to gestures with less than 200ms latency.
- **NFR-02 (Usability):** The system shall be easy to use without prior training.
- **NFR-03 (Reliability):** The system shall operate continuously without crashing.
- **NFR-04 (Maintainability):** The code shall be modular and well-commented.
- **NFR-05 (Compatibility):** The system shall run on Windows OS.
- **NFR-06 (Security):** The system shall not store or transmit video data.

3. Analysis & Design Phase

3.1 System Design Overview

The system follows a modular design where video input, gesture recognition, mouse control, and audio control are handled as separate components. MediaPipe processes hand landmarks, which are analyzed to detect gestures, and corresponding system actions are executed.

3.2 System Architecture Diagram

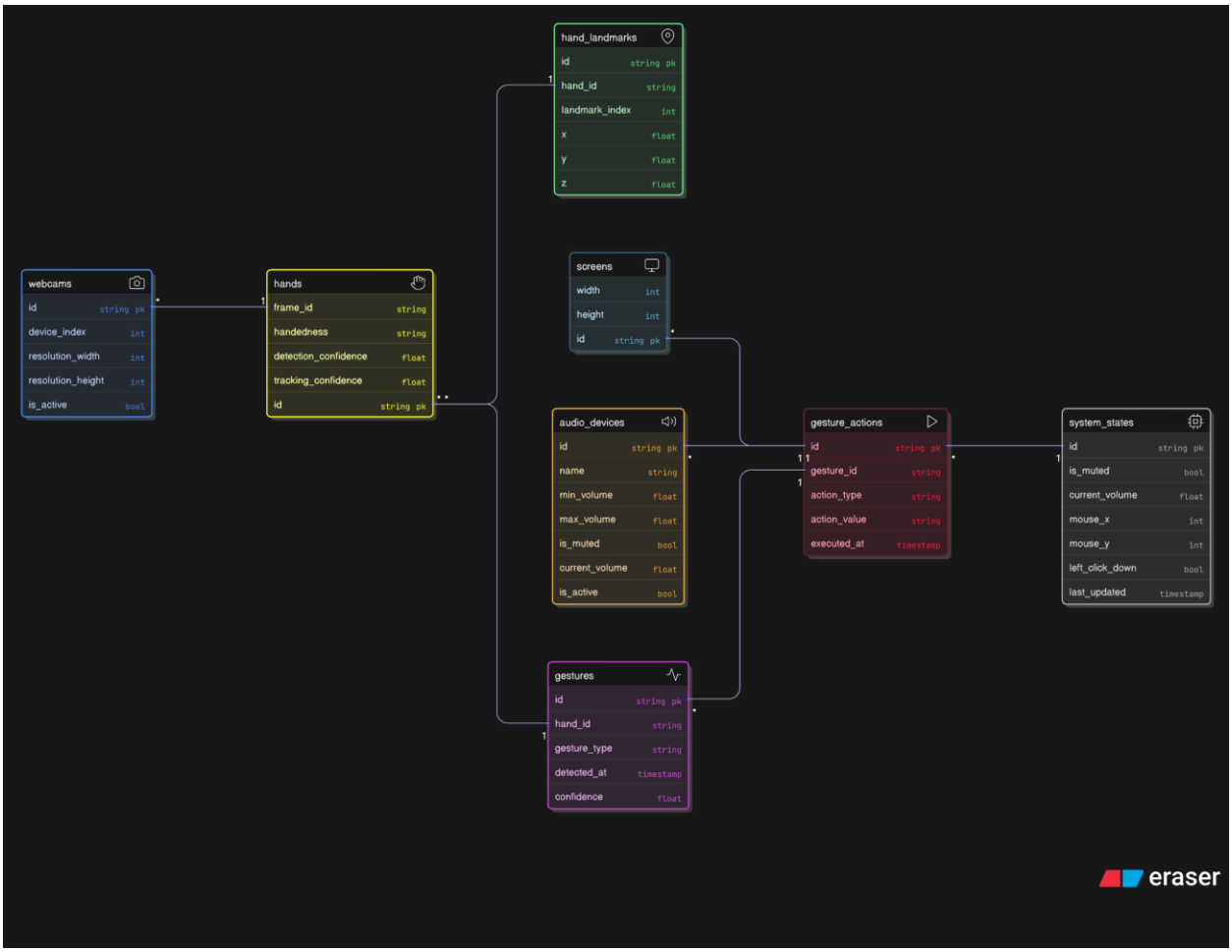


Description:

- Webcam → OpenCV (Frame Capture)
- OpenCV → MediaPipe (Hand Tracking)
- Gesture Logic Module
- Mouse Control Module (PyAutoGUI)
- Audio Control Module (PyCaw)
- UI Overlay Module

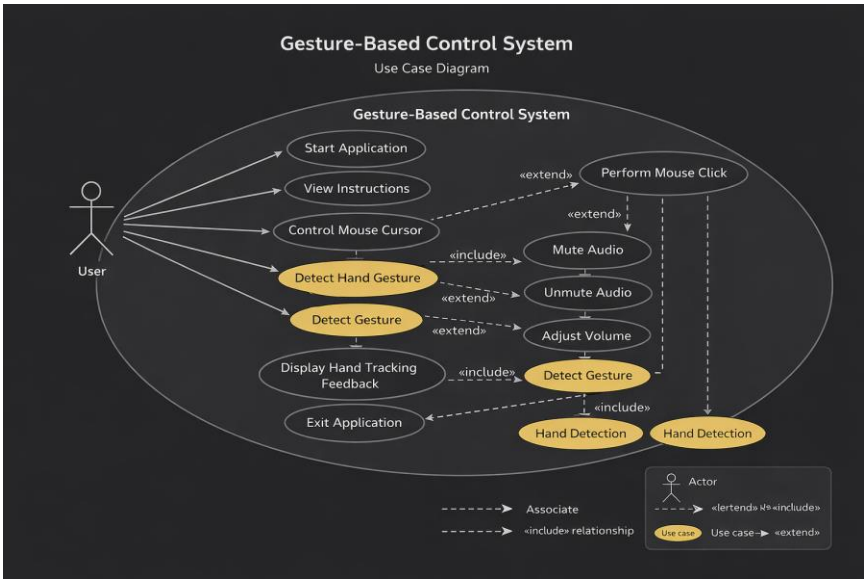
3.3 Database Design

This system does not require a database as it operates entirely in real time without persistent data storage.



3.4 UML Diagrams

3.4.1 Use Case Diagram



Actors: User

Use Cases:

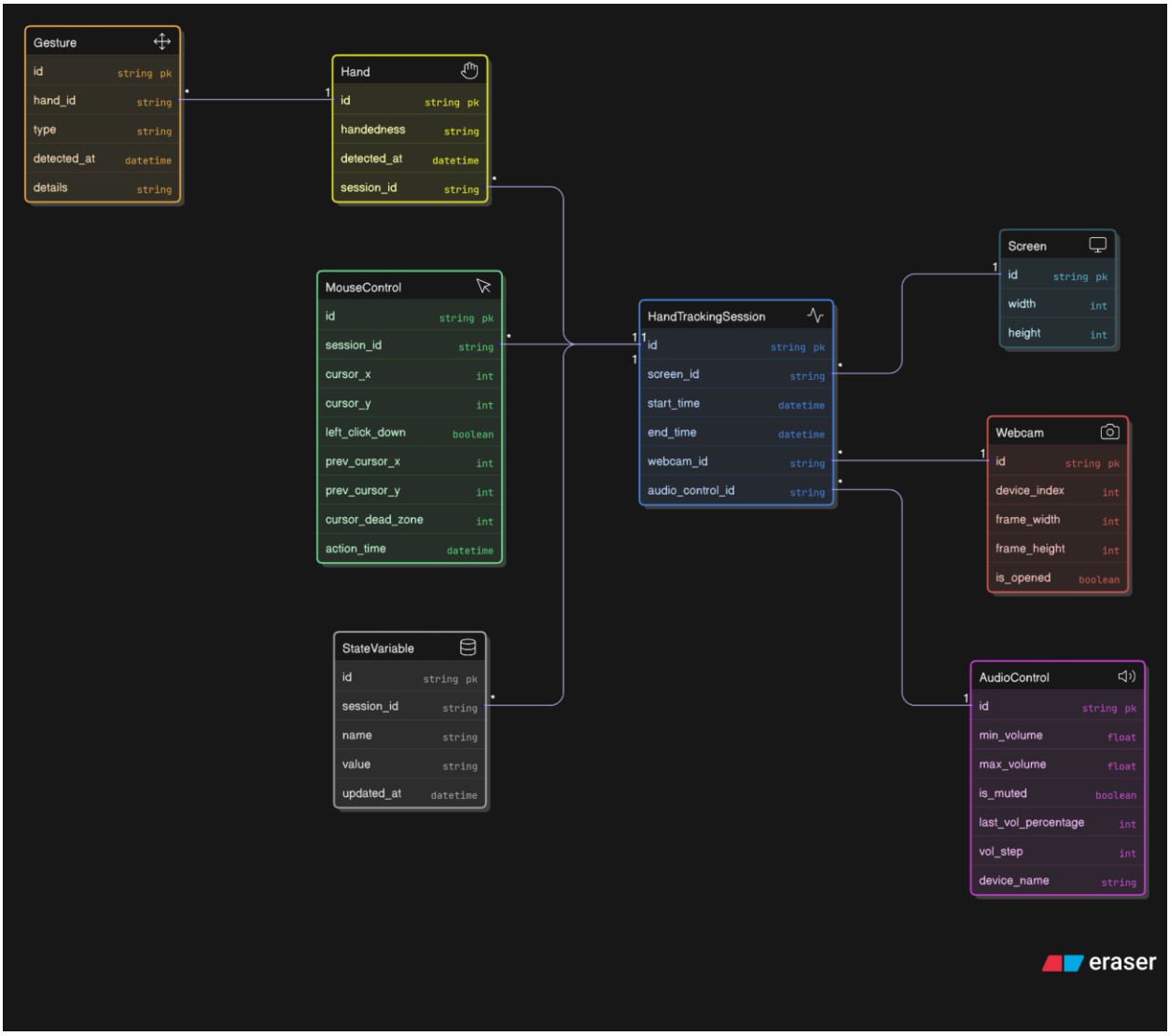
- Control Mouse
- Perform Click
- Adjust Volume
- Mute Audio
- Unmute Audio
- Exit Application

3.4.2 Use Case Descriptions

Use Case: Adjust Volume

- Actor: User
- Precondition: Webcam active
- Main Flow: User performs pinch gesture → System calculates distance → Volume adjusted
- Postcondition: System volume updated

3.4.3 Class Diagram

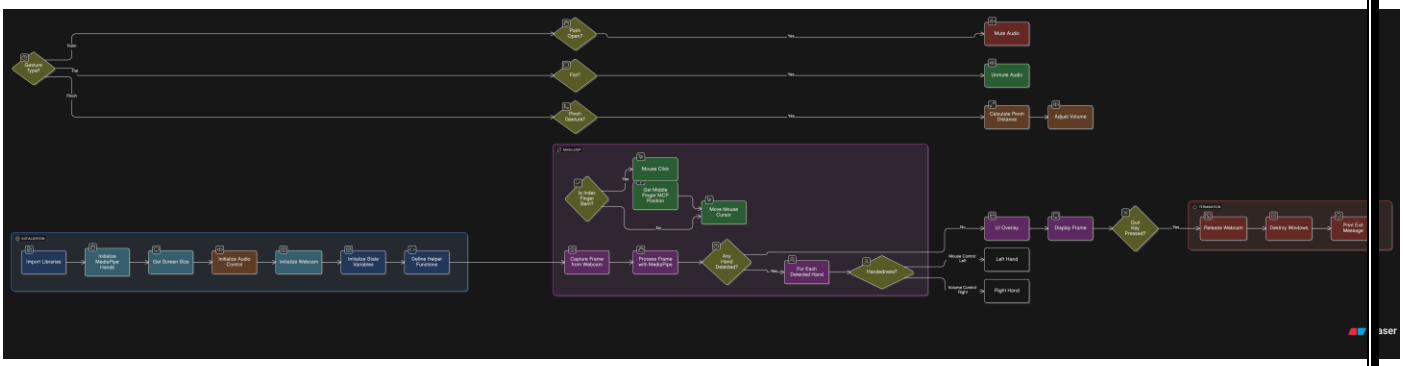


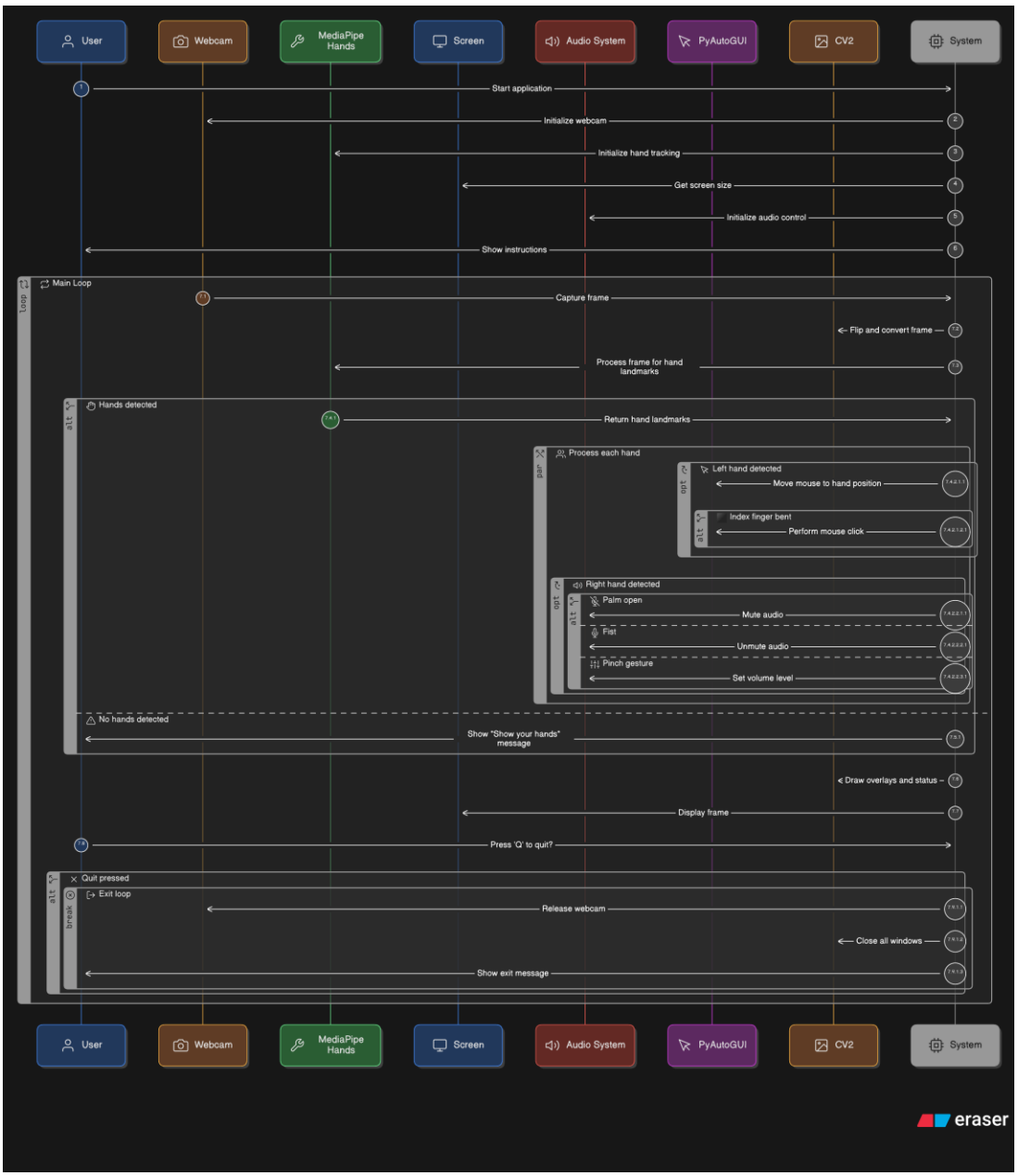
Main Classes:

- GestureController
- HandTracker
- MouseController
- VolumeController

Relationships: GestureController uses HandTracker, MouseController, and VolumeController.

3.4.4 Activity Diagrams

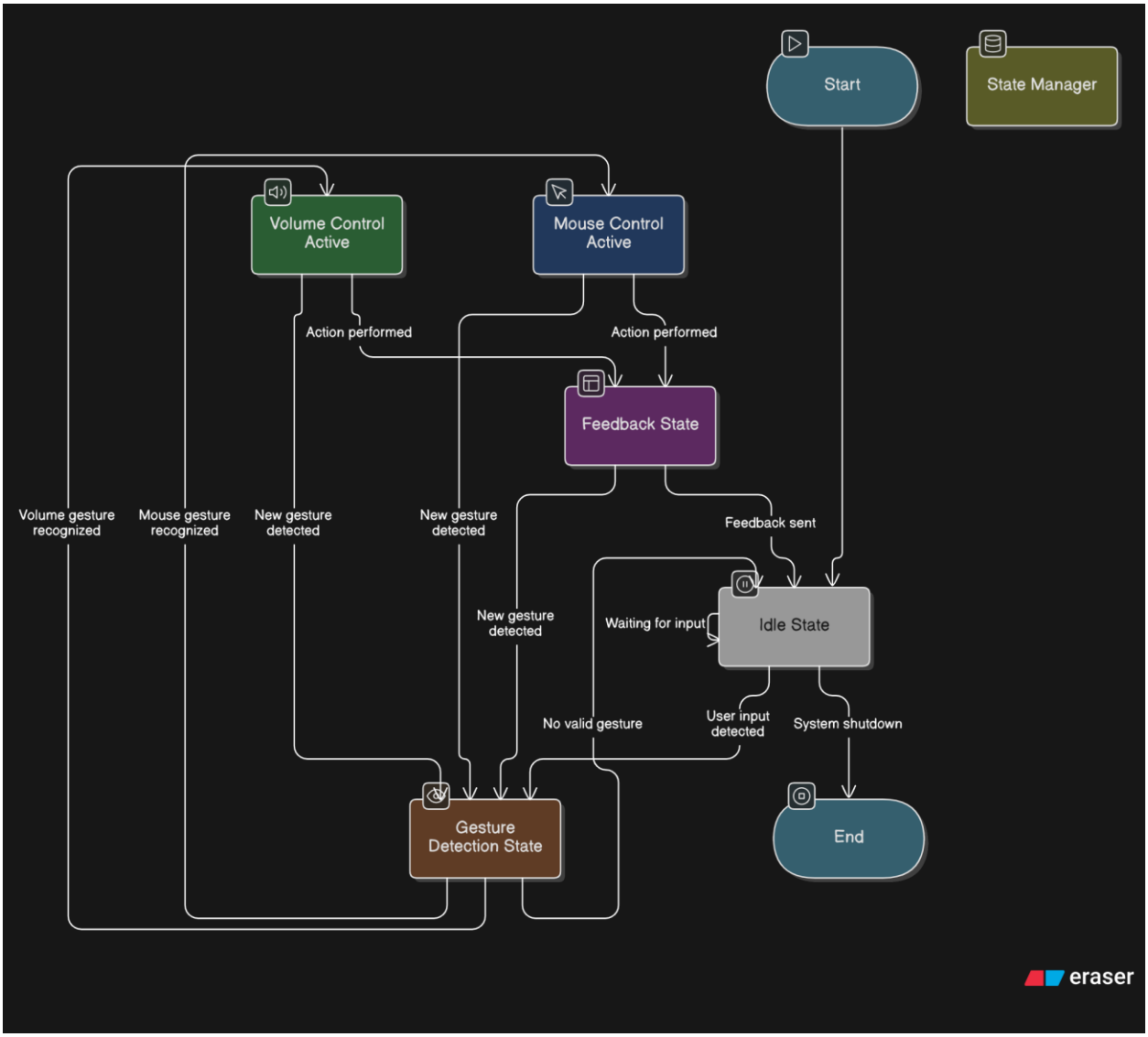




Activities include:

- Capture frame
- Detect hands
- Identify gesture
- Execute system action
- Display feedback

3.4.5 State Machine Diagram



States:

- Idle

- Hand Detected
- Gesture Recognized
- Action Executed

4. User Interface (UI) & Screen Designs

4.1 UI Overview

The Combined Gesture Control System provides a minimal, real-time visual interface that allows users to interact with mouse and system volume using hand gestures. The UI is designed to offer clear feedback, low cognitive load, and immediate system response, ensuring usability without the need for traditional input devices such as a mouse or keyboard.

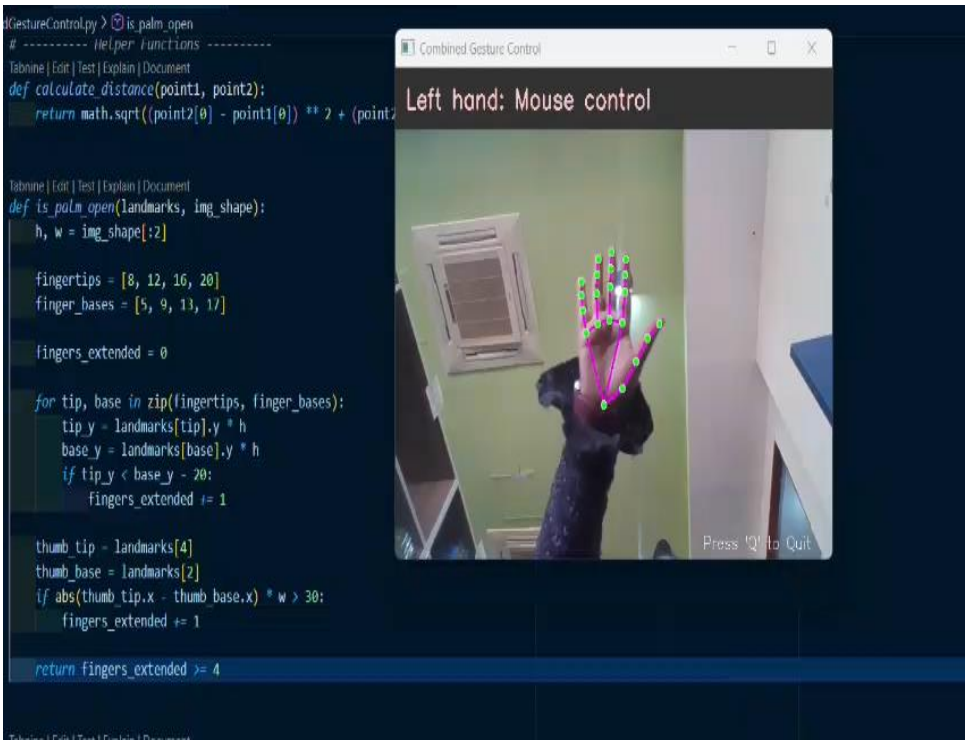
The interface operates on a single primary screen displaying live camera input augmented with gesture indicators, system status messages, and control feedback.

4.2 Wireframes / Sketches

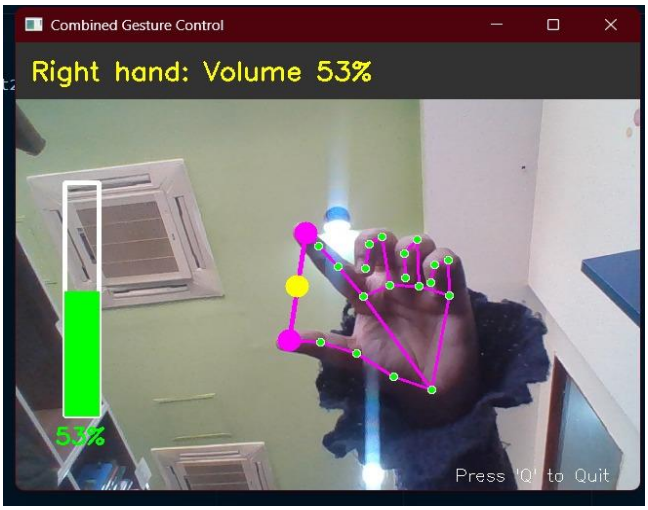
- Main camera window
- Gesture status bar at top
- Volume bar on left side

4.3 Final Interface Screenshots

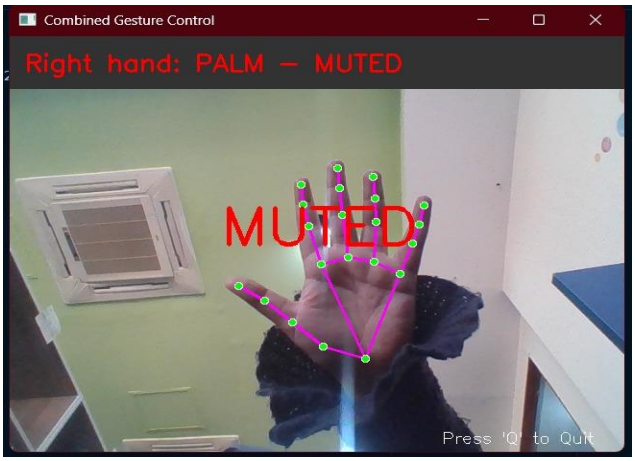
- **Main Screen:** Displays camera feed and hand landmarks.



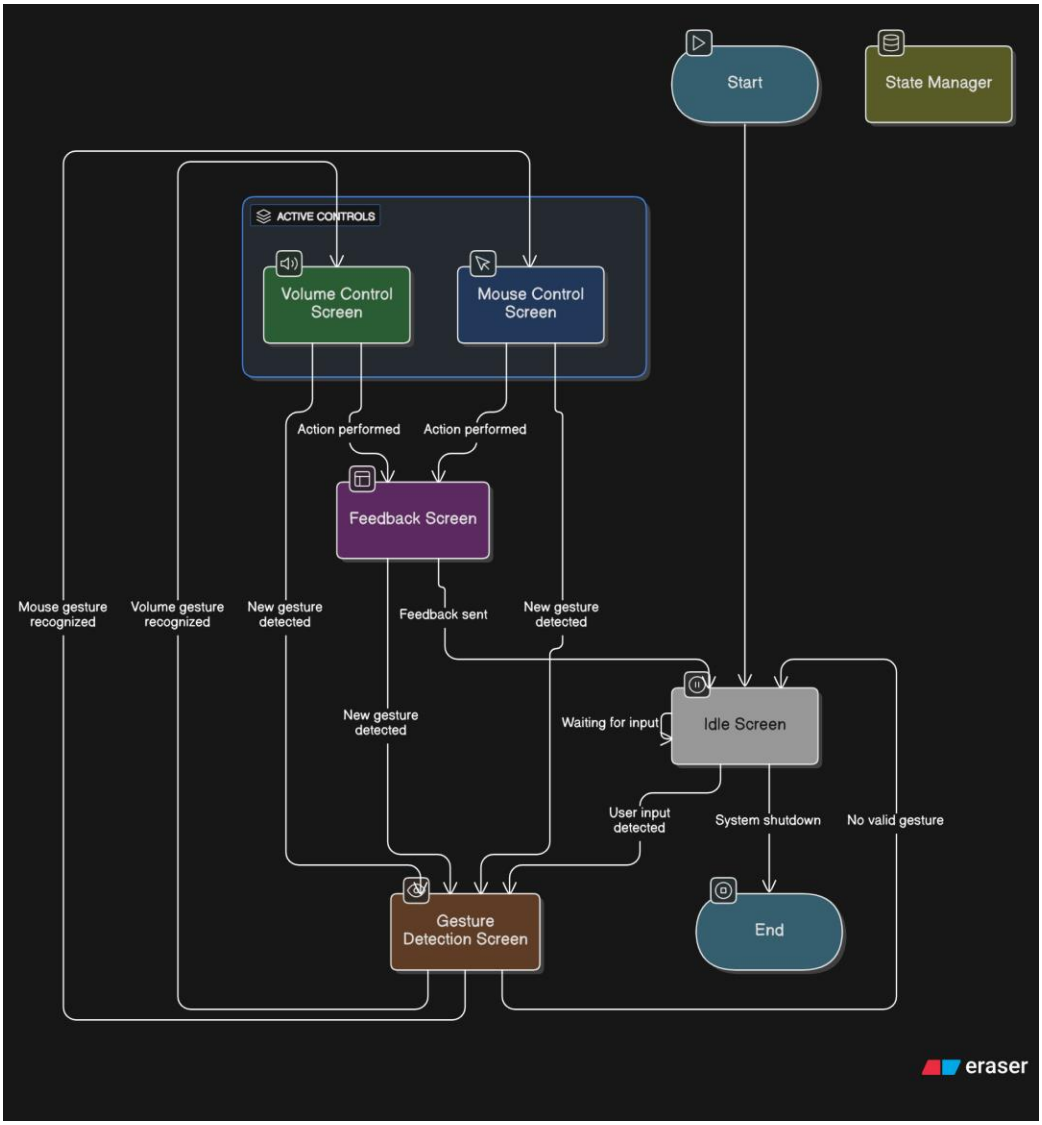
- **Volume Control Screen:** Shows volume bar and percentage.



- **Mute Screen:** Displays "MUTED" text overlay.



4.4 Navigation Flow Diagram



Single-screen application with real-time interaction loop.

5. Implementation

5.1 Technology Stack

- Programming Language: Python
- Libraries: OpenCV, MediaPipe, NumPy, PyAutoGUI, PyCaw
- IDE: VS Code
- OS: Windows

5.2 System Architecture / Folder Structure

- main.py (core logic)
- gesture_utils.py (helper functions)
- requirements.txt

5.3 Important Code Snippets

```
mp_hands = mp.solutions.hands
mp_drawing = mp.solutions.drawing_utils

hands = mp_hands.Hands(
    static_image_mode=False,
    max_num_hands=2,
    min_detection_confidence=0.7,
    min_tracking_confidence=0.7
)
print("✓ MediaPipe hand tracking initialized (2 hands)")
```

```
if results.multi_hand_landmarks:
    for idx, hand_landmarks in enumerate(results.multi_hand_landmarks):
        landmarks = hand_landmarks.landmark
```

```
mp_drawing.draw_landmarks(
    frame, hand_landmarks, mp_hands.HAND_CONNECTIONS,
    mp_drawing.DrawingSpec(color=(0, 255, 0), thickness=2, circle_radius=2),
    mp_drawing.DrawingSpec(color=(255, 0, 255), thickness=2)
)
```

```
def get_pinch_distance(landmarks, img_shape):
    """Get distance between thumb tip and index finger tip."""
    h, w = img_shape[:2]

    thumb_tip = landmarks[4]
    index_tip = landmarks[8]

    x1, y1 = int(thumb_tip.x * w), int(thumb_tip.y * h)
    x2, y2 = int(index_tip.x * w), int(index_tip.y * h)

    distance = calculate_distance((x1, y1), (x2, y2))
    return distance, (x1, y1), (x2, y2)
```

```
def calculate_distance(point1, point2):
    return math.sqrt(
        (point2[0] - point1[0]) ** 2 +
        (point2[1] - point1[1]) ** 2
    )
```

```
distance, (x1, y1), (x2, y2) = get_pinch_distance(landmarks, frame.shape)
```

```
vol_percentage = np.interp(distance, [20, 200], [0, 100])
vol_percentage = float(np.clip(vol_percentage, 0, 100))
```

```
vol_percentage_rounded = int(round(vol_percentage / vol_step) * vol_step)
```

```
vol_db = np.interp(
    vol_percentage_rounded,
    [0, 100],
    [min_vol, max_vol]
)
```

```
volume.SetMasterVolumeLevel(vol_db, None)
last_vol_percentage = vol_percentage_rounded
```

5.4 API / Module Descriptions

- **MediaPipe:** Hand landmark detection
- **PyAutoGUI:** Mouse control
- **PyCaw:** System audio control

6. Testing

6.1 Test Plan

Manual functional testing was performed to validate gesture recognition, mouse control, and volume control features.

6.2 Test Cases

Test Case ID	Description	Preconditions	Steps	Expected Result	Actual Result
TC-01	Detect hand	Webcam on	Show hand	Hand detected	Pass
TC-02	Move cursor	Left hand visible	Move hand	Cursor moves	Pass
TC-03	Left click	Index finger bent	Bend finger	Click occurs	Pass
TC-04	Pinch detection	Right hand visible	Pinch fingers	Pinch detected	Pass

TC-05	Volume increase	Pinch widen	Increase distance	Volume increases	Pass
TC-06	Volume decrease	Pinch close	Decrease distance	Volume decreases	Pass
TC-07	Mute	Open palm	Show palm	Audio muted	Pass
TC-08	Unmute	Fist	Make fist	Audio unmuted	Pass
TC-09	UI feedback	Gesture performed	Observe UI	Text displayed	Pass
TC-10	Exit app	App running	Press Q	App closes	Pass

6.3 Testing Tools Used

- Manual Testing
- OpenCV visual output

7. Deployment

7.1 Deployment Method

The application is executed locally on a Windows machine using Python.

7.2 GitHub Repository Link

https://github.com/TechnicalSamia18/handGestureControl-and-volume-Python

7.3 Live Demo / Execution Instructions

1. Install Python 3.x
2. Install dependencies using pip
3. Run python main.py
4. Ensure webcam access is enabled

8. Future Enhancements

- Support for keyboard gestures
- Custom gesture mapping
- Cross-platform support
- Improved gesture accuracy
- AI-based gesture learning

9. Conclusion

This project successfully demonstrates a real-world application of software engineering principles by delivering a gesture-based control system. The solution improves accessibility, usability, and interaction with computers using modern computer vision techniques. The project enhanced understanding of SDLC, requirements engineering, design modeling, and system implementation.

10. References

[1] Google, "MediaPipe Hands: Real-Time Hand and Finger Tracking," Google Developers, 2024. [Online]. Available: https://developers.google.com/mediapipe/solutions/vision/hand_landmarker. [Accessed: Dec. 30, 2025].

[2] OpenCV Team, "OpenCV-Python Tutorials," OpenCV Documentation, 2024. [Online]. Available: https://docs.opencv.org/master/d6/d00/tutorial_py_root.html. [Accessed: Dec. 30, 2025].

[3] A. Sweigart, "PyAutoGUI Documentation," PyPI, 2024. [Online]. Available: https://pyautogui.readthedocs.io/en/latest/. [Accessed: Dec. 30, 2025].

[4] M. B. Cox, "PyCaw: Python Core Audio Windows Library," GitHub Repository, 2024. [Online]. Available: https://github.com/AndreMiras/pycaw. [Accessed: Dec. 30, 2025].

[5] NumPy Developers, "NumPy User Guide," NumPy Documentation, 2024. [Online]. Available: https://numpy.org/doc/. [Accessed: Dec. 30, 2025].

[6] I. Sommerville, *Software Engineering*, 10th ed. Boston, MA, USA: Pearson Education, 2016.

[7] R. S. Pressman and B. R. Maxim, *Software Engineering: A Practitioner's Approach*, 9th ed. New York, NY, USA: McGraw-Hill, 2019.

[8] National University of Technology (NUTECH), "Software Engineering Course Notes," Department of Computer Science, Islamabad, Pakistan, 2025.